

CLAIMS

What is claimed is:

1 1. A method to analyze a computer program that includes a plurality of
2 blocks of code, the method comprising the steps of:
3 executing said computer program;
4 using a counter for tracking each time one of said plurality of blocks of code is
5 executed;
6 maintaining a counter cache for storing said plurality of counters of said
7 plurality of blocks of code that are most recently executed; and
8 maintaining a storage area for storing a plurality of counters of said plurality of
9 blocks of code that are not most recently executed.

1 2. The method of claim 1, further comprising the step of:
2 identifying when said counter cache is full.

1 3. The method of claim 2, further comprising the step of:
2 copying one of said plurality of counters of said plurality of blocks of code from
3 said counter cache to said storage area when said counter cache is full.

1 4. The method of claim 3, wherein the copying steps further comprises the
2 steps of:

3 determining which of said plurality of counters of said plurality of blocks of
4 code that are most recently executed is least recently executed; and
5 copying said least recently executed block of code from said counter cache to
6 said storage area when said counter cache is full.

1 5. The method of claim 3, further comprising the step of:
2 checking said code cache to determine if a block of code is being executed for
3 other than the first time; and

4 loading a counter associated with said block of code being executed for other
5 than the first time, into said counter cache.

1 6. A system for analyzing a computer program that includes a plurality of
2 blocks of code, comprising:

3 means for executing said computer program;
4 means for counting each time one of said plurality of blocks of code is executed;
5 means for maintaining a counter cache for storing said counting means of said
6 plurality of blocks of code that are most recently executed; and
7 means for maintaining a storage area for storing said counting means of said
8 plurality of blocks of code that are not most recently executed.

1 7. The system of claim 6, further comprising:
2 means for identifying when said counter cache is full.

1 8. The system of claim 7, further comprising:
2 means for copying one of said plurality of counting means of said plurality of
3 blocks of code from said counter cache to said storage area when said counter cache is
4 full..

1 9. The system of claim 8, wherein said identifying means further
2 comprises:
3 means for determining which of said plurality of counting means of said
4 plurality of blocks of code in said counter cache is least recently executed; and
5 means for copying said least recently executed block of code from said counter
6 cache to said storage area when said counter cache is full.

1 10. The system of claim 8, further comprising:
2 means for checking said code cache to determine if a block of code is being
3 executed for other than the first time; and
4 means for loading a counting means associated with said block of code being
5 executed for other than the first time, into said counter cache.

1 11. A computer readable medium for analyzing a computer program that
2 includes a plurality of blocks of code, comprising:
3 logic for executing said computer program;
4 logic for counting each time one of said plurality of blocks of code is executed;

5 logic for storing said counting logic of said plurality of blocks of code that are
 6 most recently executed; and
 7 logic for storing said counting logic of said plurality of blocks of code.

1 12. The computer readable medium of claim 11, further comprising:
 2 logic for identifying when said most recently executed storing logic is full.

1 13. The computer readable medium of claim 12, further comprising:
 2 logic for copying one of said plurality of counting logic of said plurality of
 3 blocks of code from said most recently executed storing logic to said storage logic when
 4 said most recently executed storing logic is full.

1 14. The computer readable medium of claim 13, wherein said logic for
 2 identifying further comprises:
 3 logic for determining which of said plurality of counting logic of said plurality
 4 of blocks of code in said most recently executed storing logic is least recently executed;
 5 and
 6 logic for copying said least recently executed block of code from said most
 7 recently executed storing logic to said storage logic when said most recently executed
 8 storing logic is full.

1 15. The computer readable medium of claim 13, wherein said logic for
 2 identifying further comprises:

3 logic for checking said most recently executed storing logic to determine if a
4 block of code is being executed for other than the first time; and
5 logic for loading a counting means associated with said block of code being
6 executed for other than the first time, into said most recently executed storing logic.

1 16. A system for analyzing a computer program that includes a plurality of
2 blocks of code, the system comprising:

3 a counter that tracks each time one of said plurality of blocks of code is
4 executed;

5 a counter cache that stores said plurality of counters of said plurality of blocks of
6 code that are most recently executed; and

7 a storage area that stores a plurality of counters of said plurality of blocks of
8 code that are not most recently executed code.

1 17. The system of claim 16, further comprising:

2 logic that identifies when said counter cache is full.

1 18. The system of claim 17, wherein said logic copies one of said plurality of
2 counters of said plurality of blocks of code from said counter cache to said storage area
3 when said counter cache is full.

1 19. The system of claim 17, wherein said logic determines which of said
2 plurality of counters of said plurality of blocks of code in counter cache is least recently

3 executed, and copies one of said plurality of counters of said plurality of blocks of code
4 from said counter cache to said storage area when said counter cache is full.

1 20. The system of claim 17, wherein said logic checks said code cache to
2 determine if a block of code is being executed for other than the first time, and loads a
3 counter associated with said block of code being executed for other than the first time,
4 into said counter cache.